2. 2. 2005

1 Je dany binarny strom v reprezentacii vhodnej na pouzitie techniky Euler tour. Najdite EREW PRAM algoritmus, ktory v case O(log n) a praci O(n) vypocita jeho diameter.

2 Je dane cislo e>0. Najdite common CRCW PRAM algoritmus, ktory vyrata maximum z n prvkov v konstantnom case a praci n^{1+e}.

3 Nech S1 a S2 su dve mnoziny bodov v rovine, pricom n=|S1|+|S2|. Napiste algoritmus, ktory v case O(log n) a praci O(n log n) zisti, ci existuje priamka, ktora oddeluje S1 od S2. Mohli sme predpokladat, ze S1 a S2 su disjunktne.

4 Je dany n-vrcholovy strom tak, ze pre kazdy vrchol je dany jeho prvy syn a nasledujuci surodenec. Najdite CREW PRAM algoritmus, ktory v case O(log n) a v praci O(n) vytvori pole obsahujuce vsetky listy stromu v poradi zlava doprava. Mohli sme predpokladat, ze strom je zakoreneny a koren je 1. prvok postupnosti.

---

18. 1. 2005

1 Mame pole n celych cisel A = (a_1,..,a_n), pricom a_i je ofarbene farbou l_i \in {1,2,...,k}, kde k <= log n. Napiste algoritmus, ktory najde prefixove sucty vsetkych cisel danej farby l, v poradi v akom sa vyskytuju v poli A, t.j. na i-tej pozicii bude vypocitana hodnota $\sum_{j \text{ in } \{1,\dots,i\} \;\&\; lj = li} a_j$ pre vsetky farby l v case O(log n) a s linearnym poctom operacii. Mame pouzit EREW PRAM.

Trivialne riesenie:

Budeme mat dvojrozmerne pole A - pre kazdu farbu a kazde policko zo vstupneho pola bude tu zaznam. Zapiseme na A[i,j] = 1 ak a_j je farby i a A[i,j] = 0 ak a_j nie je farby i. Ked toto mame spocitame prefixove sumy v kazdom poli paralelne. Cas bude O(log n) (lebo pre kazdu farbu mame pole dlzky n) ale pocet operaci bude O(n log n) lebo v kazdom poli urobime pracu O(n), co dokopy urobi pracu O(n log n), co je prilis vela.

Skusime teda pouzit staru znamu fintu a zlepsime to:

Vstupne pole velkosti n si rozsekame na kusy velkosti log n (bude ich n/log n) a v nich sekvencne napocitame prefixove sumy po jednotlivych farbach.Toto nam nic nepokazi, cas tejto operacie je O(log n) a praca bude O(n). Vezmeme si z kazdeho takehoto logaritmickeho chlievu pre kazdu farbu prefixovu sumu, co sme spocitali a zapiseme si ju do pola A.

Budeme mat A pre log n farieb a pre n/log n zaznamov. Pre kazdy chlievik budeme mat pre kazdu farbu zapisanu prefixovu sumu co sme spocitali. Teda este raz:

A[i,j] = p je v j-tom chlieviku prefixova suma, ktoru sme tu nascitali pre farbu i je p. Tento zapis vie urobit napr O(n/log n) procesorov v case O(log n).

Potom nascitame prefixove sumy v poli A pre kazdu farbu. Cas to bude trvat O(log (n/log n)) = O(log n) a praca bude O(n/log n) = O(n). V poli A[i,j] bude nascitane prefixova suma pre prvky v chlievikoch mensich ako j pre danu farbu i. Teraz na spojenie pre kazdu farbu treba vziat prefixovu sumu toho zvysku predtym pred chlievikom (je v poli A) a pripocitat ju ku kazdej sekvencne vypocitanej prefixovej sume v chlieviku. Toto staci urobit sekvencne aby sme neporusili EREW.

2 Je dana postupnost {a_1,...,a_n} a_i \in R. Napiste optimalny common CRCW PRAM algoritmus, ktory v case O(1) zisti, ci je postupnost bitonicka. (ak sa nam hodi, mozeme predpokladat, ze tam nie su dve rovnake prvky)

3 Mame dany kruh pozostavajuci z vrcholov 1 az n, t.j. mame dane pole V dlzky n, kde hodnota V[i] urcuje praveho suseda vrchola i na kruhu. Dalej mame danu mnozinu hran E taku, ze s kazdym vrcholom je incidentna nanajvys jedna hrana z E, t.j. mame dane polia E_1 a E_2 dlzky l, kde i-ta hrana z E vedie medzi vrcholmi E_1[i] a E_2[i].

Najdite EREW PRAM algoritmus, ktory v case O(log n) a praci O(n) zisti, ci je mozne nakreslit vsetky hrany E vo vnutri kruhu tak, aby sa ziadne dve nepretinali.

Mame kruh reprezentovany ako spajany zoznam. Vyberieme si vrchol a tam tento zacykleny zoznam prerusime. Dostaneme v podstate spajany zoznam. Urobime na nom list ranking a ten nam zisti vzdialenost kazdeho vrcholu od konca. Takze si tie vrcholy precislujeme. Spolu s nimi si precislujeme aj hrany aby sa nam zachovali. To sa da urobit asi v list rankingu alebo to urobime tak, ze urobime len list ranking, zapametame si stare aj nove cisla a potom si pozrieme do E_1 a E_2 ze ako ide hrana a zmenime ju podla noveho ohodnotenia. Toto vsetko zatial vieme urobit na EREW v case O(log n) s O(n) procesormi.

Teraz mame uz problem pretransformovany. Vznikol nam kruh, kde susedia prvku i su prvky i-1 a i+1. Z kazdeho vrcholu na tomto kruhu ide najviac jedna hrana. Ideme zistit, ci sa hrany pretinaju. To zistime takto:

Ked ide hrana z vrcholu i do vrcholu j, zapiseme si interval <i,j> (i< j).

Toto urobime pre vsetky hrany. Kedze ideme z kazdeho vrcholu ide najviac jedna hrana, taketo intervaly nenaju zaciatocny a koncovy bod rovnaky. Ak sa dve hrany - napr ab a cd (a<b c<d) krizia tak <a,b> \cap <c,d> = <c,b> (nakreslite si to). To ci <a,b> prienik <c,d> = <c,b> vieme zistit prefixovymi sumami. Za zaciatok intervalu dame +1 za koniec -1 spocitame prefixove sumy a pozrieme sa ci zaciatok a koniec intervalu maju rovnaku prefixovu sumu a je to. Toto tiez vieme urobit na EREW v case O(log n) a v praci O(n).

4 Je dana mnozina S, ktora pozostava z n bodov v rovine. Dalej je dany bod p, ktory nepatri do S. Voronoiov mnohouholnik V(p) je mnohouholnik, ktory ohranicuje vsetky body q, ktore su blizsie k p ako k lubovolnemu bodu z S.

Najdite CREW PRAM algoritmus pracujuci v case O(log n) a praci O(n log n), ktory pre danu S a p najde Voronoiov mnohouholnik V(p).

Majme bod A z danych n bodov a bod P. Body, ktore su blizsie k bodu P lezia v rovine, ktoru urcime nasledovne:

Oznacme si stred medzi A a P ako S, S = (A+P)/2

Vezmime si smernicu priamky AP a jej normalu. Jej normala bude smernica kolmej priamky, ktora prechadza bodom S a ktora urcuje hranicu - jedna polrovnina je mnozina bodov blizsich k P. Staci vziat vsetky taketo polroviny a ich prienik je to, co hladame. Ten je uz v Jajovi opisany ako sa robi. To je asi vsetko.

Jeden priklad sa vzdy opakuje z tych predchadzajucich (2.) a jeden priklad je vzdy len na acelerate cascading (1.) teda vymysli sa riesenie, ktore ma zlu pracu a potom sa do toho mlati standardnymi postupmi aby praca bola dobra. Ostatne dva su vzdy na to, ze ich treba previest na nieco, co uz je v Jajovi vyriesene.

5. 1. 2005

1 Nech M je pole dlzky n, ktore obsahuje iba cisla 0 alebo 1. Je dane cislo k < n. Najdite EREW PRAM algoritmus pracujuci v case O(log k) a praci O(n), ktory vypocita pole M' dlzky n-k, pre ktore plati:

M'[i] = 1 <=> (\forall j \in {i,...,i+k}) M[j] = 1

Cize chceme vypocitat pole M', ktore bude obsahovat na i-tom mieste 1 prave vtedy, ked na poziciach {i, i+1, ..., i+k} povodneho pola M su 1. Pozicie {i, i+1, ... i+k} tvoria suvisly usek, co vyuzijeme.

Rozdelme si povodne pole M na suvisle useky dlzky k, posledny usek pripadne doplnime nulami. Vsimnime si teraz jedno taketo pole dlzky k. Klasickym algoritmom pre EREW v case O(log DLZKA) a praci O(DLZKA) v nom vypocitame prefixove a sufixove sumy s operaciou nasobenia. Toto nam teda trva cas O(log k) a pracu O(k). Co keby sme to spravili pre vsetky polia paralelne? Cas zostane O(log k), celkova praca bude O(n).

Uvedomme si, co nam tieto prefixove a sufixove sumy hovoria. Ak napr. je nejaka sufixova suma 1, znamena to, ze vsetky prvky toho sufixu su 1 (pretoze ak v nasobni je co i len jeden prvok 0, vysledok je 0).

Ako vypocitame pole M', inak povedane -> ako teraz pre nejaku poziciu i zistime ci pozicie i, ..., i+k obsahovali v povodnom poli same jednotky?

Suvisla oblast {i, i+1, ..., i+k} je dlha k+1, teda sa rozklada cez prave dve nase mensie podpolia dlzky k. A vsimnime si, ze tychto k+1 prvkov tvori nejaky prefix v prvom podpoli a nejaky sufix v druhom podpoli. Avsak pre kazdy prefix aj sufix uz mame vypocitane, ci ho tvoria same jednotky.

Algoritmus 1.:

a) *predpocitame prefixove a sufixove sumy suvislych podpoli dlzky k. prvok*
b) *M'[i] urcime vysetrenim dvoch hodnot, prislusnej prefixovej a sufixovej sumy takej, ze zlozenim prislusneho prefixu a sufixu dostaneme prvky {i, i+1, ..., i+k}.*

Myslim, ze na tento priklad stacilo sikovne pouzitie klasickeho algoritmu na prefixove sumy.

2 Dane je pole A = (a_1, ..., a_n), kde a_i su prirodzene. Nech d_i oznacuje dlzku najdlhsej suvislej neklesajucej podpostupnosti A zacinajucej od indexu i (tj. d_i = max{j | a_i <= ... <= a_{i+j-1}). Najdite optimalny EREW PRAM algoritmus, ktory v case O(log n) vyrata pole (d_1, ..., d_n). Napiste program pre konkretny procesor p_i.

Vsimnime si, ze zasa po nas chcu EREW a zasa cas O(log n). To znamena, ze zasa to bude nieco na sposob prefixovych sum... myslim tym, nejake ratanie v binarnom strome najskor od listov ku korenu a potom od korena naspat k listom.

Obmedzime sa na vyvazeny binarny strom. Teda kazdy vrchol bude reprezentovat suvisly interval vstupu (listy v danom podstrome). Este nevieme co budeme pocitat, ale pokial to budeme robit "lokalne", teda na vypocet nejakej hodnoty vo vrchole pouzijeme najviac hodnoty jej deti (klasicky vypocet po vrstvach), tak mame zarucene, ze to pojde na EREW modeli a v case O(log n) a praci O(n).

Druhe pozorovanie je, ze ich zaujima SUVISLA podpostupnst. To znamena, ze si vo vrcholoch musime pocitat NIECO take, aby sme z hodnot deti, vedeli sikovne zistit hodnoty v otcovi.

A teraz, ake hodnoty si vlastne chceme vo vrchole pamatat?

Nech vo vrchole v je A[v] index najlavejsieho listu podstromu v a B[v] najpravejsieho. Teda v zmysle nasich intervalov, podstrom vrcholu v reprezentuje interval vstupu A[v], B[v].

Hodnoty A[v], B[v] sa pocitaju vo faze od "listov ku korenu" trivialne. A kedze nas algortimus potrebuje vyprodukovat aj vysledok "dlzku najdlhsej suvislej neklesajucej podpostupnosti", tak si ho rovno pre kazdy vrchol vypocitajme.

Nech teda hodnota C[v] znaci pre vrchol v dlzku najdlhsej neklesajucej podpostunosti pola X pocinajuc indexom A[v].

Algoritmus 2.:

a) *V listoch inicializujeme hodnoty (pre i-ty list) A=i, B=i, C=1.*
b) *postupujuc od listov ku korenu pocitame hodnoty A, B, C. Hodnoty A a B pocitame trivialne. Ako vypocitame hodnotu C[v], ked vieme hodnoty pre jeho potomkov u a w?*
   1. *Ak C[u] pokryva cely interval A[u]...B[u], teda postupnost je neklesajuca na celom intervale A[u]...B[u], tak je ista mosnost, ze bude pokracovat aj v intervale A[w]...B[w] (uvedomme si, ze A[u]...B[u]A[w]...B[w] = A[v]...B[v]). To ci pokracuje, zistime tak, ze hodnota X[B[u]]<=X[A[w]]. Ak pokracuje, tak C[v]=C[u]+C[w], inak C[v]=C[u].*
   2. *Ak C[u] nepokryva cely interval A[u]...B[u], tak zjavne C[v]=C[u].*
c) *postupujuc od korena k listom prepocitame hodnoty C tak, aby zahrnovali aj podpostupnosti dlhsie ako interval A[v]...B[v]. Toto je podobne rozobratie moznosti ako v pripade b).*

Nakoniec mame v listoch hodnoty C, ktore potrebujeme.

Tento priklad bol znovu na jednoduchu modifikaciu klasickeho postupu "od listov ku korenu" a "od korenu k listom" na vyvazenych binarnych stromoch.

$\boxed{3}$ V rovine je danych n obdlznikov, ktorych hrany su rovnobezne so suradnicovymi osami (polia X, Y, A, B obsahuju v poradi ich suradnice laveho dolneho rohu, dlzku a vysku). Najdite PRAM algoritmus, ktory v case O(log n) a praci O(n^2) najde plochu ich zjednotenia.

Priklad riesime presne rovnakym sposobom ako jeho sekvencne riesenie, len ho prepiseme efektivne-paralelne.

Predpokladajme, ze su vsetky suradnice rozne. Najskor si koncove body obdlznikov utriedime podla x-suradnice, cim nam vzniknu "pasiky" tvorene susednymi x-suradnicami v uriedenom poradi x-suradnic, take, ze vnutri tychto pasikov sa nenachadzaju ziadne dalsie body.

Kazdy takyto pasik sa da spracovavat sucasne a teda problem spociva vo vypocitani plochy zjednotenia v jednom takomto pasiku. (Vysledne hodnoty pre pasiky sa spocitaju do jednej premennej klasickym algoritmom na prefixove sumy, co nam cas ani pracu nepokazi).

Az do konca sa teda budeme zaoberat tym, ako vypocitame, plochu zjednotenia obdlznikov v konkretnom x-pasiku.

No ak si vsimneme, tak mozeme vsetky obdlzniky rozdelit aj do y-pasikov (podla ich y-suradnic vrcholov). Pre kazdy x-pasik je potom dolezite, ktore y-pasiky su v nom pokryte a ktore nie.

Nato aby sme veci vedeli sikovne spocitat si este vopred utriedime koncove body aj podla y-suradnic. Toto pole vyuzijeme potom pre vypocet v (kazdom) x-pasiku nato, aby sme si pre kazdy obdlznik (ktory do prislusneho x-pasiku patri) zapisali do pola P dlzky 2n (kazdy x-pasik ma vlastne pole P):

ak prislusny obdlznik zasahuje do x-pasiku:

+1 na index (v utriedenom poradi y-suradnic) tam kde obdlznik zacina

-1 na index (v utriedenom poradi y-suradnic) tam kde obdlznik konci

ak prislusne obdlznik nezasahuje do x-pasika tak na obe miesta (v utriedenom y-poradi) zapiseme 0.

Pole P vieme generovat n procesormi (kazdi pre jeden obdlznik) pre jeden x-pasik v case O(1) a praci O(n).

V tomto poli P dlzky 2n (pre kazdy obdlznik mame vyhradene dve miesta) obsahujucom +1,-1,0 nam prefixove sumy budu znamenat, ci je kokretny y-pasik v danom x-pasiku pokryty. Vypocitat prefixove sumy pre x-pasik a vysledne plochy zratat (znovu alg. na prefixove sumy) na trva pre jeden x-pasik cas O(log n) pracu O(n).

Kedze spracuvame n pasikov paralelne, dostavame cas O(log n) a pracu O(n^2).

Priklad v podstate spocival v prepisani sekvencneho postupu do paralelneho sveta.

$\boxed{4}$ Maticou susednosti je dany n-vrcholovy graf G. Najdite PRAM algoritmus, ktory v case O($\log^2 n$) zisti, ci je graf G bipartitny.

Tento priklad bol mozno trochu trikovy. Aspon podla mna. Spominany trik spocival v najdeni kostry T. Kostru sme nasli pouzitim "algoritmu z prednasky". Uvazujme suvisly graf, pretoze nesuvisly sa vyriesi jednoducho po komponentoch.

Potom si staci vsimnut, ze kostra T je strom, ktory obsahuje vsetky vrcholy G.

Teda ak G je bipartitny, tak existuje take rozdelenie vrcholov do dvoch mnozin, ze nejdu hrany medzi tymito dvoma mnozinami. Ako to ale pre taketo bipartitne rozdelenie vyzera v kostre T?

Treba si uvedomit, ze strom ma len dve mozne taketo rozdelenia, ktore su este k tomu "dualne", cize staci uvazovat jedno taketo rozdelenie. Jedno taketo rozdelenie najdeme napriklad tak, ze strom zakorenime a urcime level - vzdialenost od korena a vrcholy ofarbime farbami level[v]%2.

Takto dostavame jedinu biparticiu vrcholov stromu T, co ale je zaroven jedina mozna biparticia vrcholov grafu G a teda nam staci paralelne v jednom kroku (jeden procesor pre kazdu hranu) zistit, ci jej koncove vrcholy maju rovnaku farbu.

Ak pre kazdu hranu maju vrcholy roznu farbu <- nasli sme platnu biparticiu pre G, inak G nie je bipartitny, pretoze ofarbenie vrcholov je jedine mozne (az na to "dualne" ze vymenime farby presne naopak, cim sa vsak bipartitnost G nezmeni).

---

21. 12. 2004

1 Mame nezakoreneny strom T zadany cyklickym zoznamom incidentnych hran a pre kazdu hranu aj pointer na opacnu hranu (=klasicky). Najdite algoritmus v case O(log n) a v praci O(n) na najdenie 2-farbenia T.

Z jaju sa odkazes, ze vies zakorenit strom a zratat level l_v (vzdialenost od korena) kazdemu vrcholu v. potom vrcholu v priradis farbu l_v modulo 2. = 4 vety:).

2 Mame v rovine N obdlznikov s hranami rovnobeznymi so suradnicovymi osami. najdite CREW PRAM algoritmus v case O(log n) a v praci O(n^2), ktory zrata plochu ich zjednotenia.

Naznak riesenia, ktore treba domysliet: nazvime eventmi jednotlive vrcholy obdlznikov a aj vsetky priesecniky hran (dokopy by ich malo byt O(N2)). Usortime eventy najprv podla x-ovej potom podla y-ovej suradnice do pola B. Tieto eventy nam rozdelia plochu na zvisle "pasiky", v ktorych nie je ziadny event. V poli B je kazdy "pasik" urceny suvislou postupnostou eventov s rovnakou x-ovou suradnicou. najdeme "prechody" medzi pasikmi v poli B. Potom pre kazdy pasik paralelne: kazdej vrchnej hrane obdlznika priradime +1. kazdej spodnej -1. zratame prefix-sumy, co nam da nove pole (nazvime ho C). na i-tom mieste v C je kladne cislo prave vtedy, ked je k nemu zodpovedajuca cast cela pokryta aspon jednym obdlznikom, inak je cela nepokryta. teda pre kazdu pokrytu cast zratame jej obsah (v O(1)). (koniec odseku "pre kazdy pasik"). potom musime znamym algoritmom z Jaju zratat celkovu sumu tychto O(n^2) casti. detaily nechavam na DU.

3 Je dana postupnost (a_1, ... ,a_n), a_i \in R. Napiste optimalny common CRCW PRAM algoritmus, ktory v case O(1) zisti, ci je postupnost bitonicka.

Postupnost je bitonicka <=> ma max. 2 extremy. teda vytvorime druhe pole B, kde na i-tom mieste je 1 <=> v a_i je extrem. Teraz 3krat zistime najlavejsiu jednotku v B a odstranime ju - toto pravdepodobne vacsina z nas mala v nejakej DU. Teda vieme zistit, ci su v poli B aspon 3 jednotky. Ak su=>postupnost nie je bitonicka, inak je.

4 Mame pole N celych cisel A=(a_1, ..., a_n), pricom a_i je ofarbene farbou l_i \in {1,...,k}, kde k <= log n. Napiste optimalny common CRCW PRAM algoritmus, ktory v case O(log log n) najde pre kazdu farbu minimum z tych a_i, pre ktore l_i=l

Najprv trivialne riesenie: minimum pola vieme zistit v case O(log log n) a v praci O(N). Teda vytvorime si log n poli B_i - pre kazdu farbu jedno. Vsetky maju N prvkov. Ak a_i je zafarbene farbou j, tak B[i][j]=a_i, inak B[i][j]=INFINITY. na kazdom poli zratame minimum a mame algoritmus s pracou O(N log n) a s casom O(log log n). Tradicna finta: rozdelme A na useky dlzky log n. V kazdom z nich zratajme potrebne informacie v case O(log log n) a v linearnej praci (nizsie vysvetlim ako), potom mame log n poli kazde velkosti N/log n a pustime na to nas neoptimalny algoritmus.

Na to aby, sme pre kazdy usek dlzky log n zratali to co chceme, staci nam algortmus s linearnou pracou a s casom log n. potom tento pusteny na dlzku log n bude mat cas log log n.

takze najprv algoritmus s casom O(log n) a pracou O(n log n): vybudujme uplny vyvazeny binarny strom nad polom(ako v prefixsumach). zratajme nas problem v kazdom uzle. na to nam staci sa pozriet na udaje z jeho 2 deti a v case O(1) a v praci O(log n) vieme zistit "udaje" v nom. Udajmi myslim tabulku minim pre kazdu farbu :) . podrobnosti si rozmyslite.

este ho zlepsime na pracu O(n): tradicna finta: rozdelme pole na useky dlzky log n. v kazdom z nich zratame problem sekvencne a na vysledok pustime neoptimalny algoritmus.

---

30. 7. 2004

1 Majme pole prirodzenych cisiel A[1..n]. Najdite alg. na commonCRCW, taky, ze nam vypocita pole L[l_1,..,l_n] v case T=O(1) a praci W=O(n^2). Pricom l_i je definovana takto: l_i = max { j | a_j = a_i, j < i }

2 Majme n useciek v rovine, spravme si priemet do X-ovej osi. Chceme zratat dlzku tohto priemetu (s vynechanim dier ak su) v T=O(log(n)), W=O(n*log(n))

3 Majme komparatorove siete s n vstupmi. Dokazte, alebo vyvratte: Siet je triediaca prave vtedy ked triedi vsetky vstupne vektory z 0, 1 (\all vektor x \in {0, 1}^{n})

4 Pre i-ty prvok pola A[1..n] definujeme farbu funkciou c(i) <= k <= log(n). Chceme zratat prefixove sumy po jednotlivych farbach.

5 Funkcia D (pole?) definuje les, chceme zratat pre kazdy vrchol hodnotu minimalneho vrcholu v jeho strome.

---

11. 6. 2003

Najdite vsetky mosty v grafe v O(n^2) O($\log^2 n$)

idea: najst kostru grafu (obsahuje vsetky mosty) cim sa zmensi mnozina potencionalnch mostov, spominana zlozitost je presne zlozitost najdenia kostry (presne si asi nepametam tu zlozitost) Potom nejak pouzit kontrahovanie stromu (ako pri hladani kostry)

Pole prvov A(a_1, ... a_n) ofarbene farbami 1.. log n. Zratajte prefixove sumy pre vsetky farby. V case O(log n) a operaciami O(n)

takze pomocou rekurzie, delite pole na pol a pri spatnom prechadzanie konstrujete okrem sum aj pole dlzky logn kde su indexy najpravejsich vyskytov prvkov zafarbenych jednotlivymi farbami. Pri ratani sumy pricitujete k prvokm z druhej polovicky prvky na ktore ukazuje to to pole podla farby prvku. Samozrejme praca nie je optimalna, preto si treba pole predspracovat pokym nebude mat n/log n dlzku a potom na topustit popisanu zverinu

Mame pole celych cisel A=(a_1,...,a_n) pricom a_i je ofarbena farbou l_i in {1,2,...,k}, kde k <= log n. Napiste algoritmus, ktory najde prefixove sucty vsetkych cisel danej farby l (v poradi, v akom sa vyskytuju v poli A) pre vsetky farby l v case log n a s linearnym poctom operacii.

Majme binarny vyhladavaci strom T(tj. kazdy vrchol je bud list, alebo ma prave dvoch synov). S kazdym vrcholom si pamatame jeho rodica p(v) a surodenca s(v). Najdite optimalny algoritmus, ktory v case O(log n) najde minimalne pokrytie stromu T (t.j. mnozinu vsetkych vrcholov takych, ze kazda hrana je implementovanaaspon s jednym z nich)

Mame pole n celych cisel A=(a_1,..,a_n), pricom a_i je ofarbene farbou l_i in {1,2,..,k}, kde k <= log n. Napiste algoritmus, ktory v case O(log log n) najde pre kazdu farbu l minimum z tych a_i, pre ktore l_i=l. Snazte sa o co najmensi pocet operacii.

Priblizne riesenie:

ako v DLDT, len narezat n cisel na useky o nieco dlhsie ako odmoc(n). Kazdu cast optim.sekv.algoritmom vyriesit, pamatat si este kratsie pole farieb vo vrcholoch (vsetkych?). Po vyrieseni tych casti (=podstromov) mam v kazdom jej "koreni" (je ich n/odmoc(n) = odmoc(n)), presnejsie v tomto jeho poli farieb, minima z tych farieb.

Teraz chcem urobit z nich vsetkych minimum. Pri klasicom ratani minima tam je nieco vysoke, (pocet procesorov?,cas?) preto to chceme znizit, a to tak, ze ten vstup na zaciatku rozdelime na useky odmoc(n * log n) a usekov bude menej.

To by malo stacit, aby sa vypocet minima zmestil do log n (alebo log log n?).

Mame dane pole lavych a pravych zatvoriek. Napiste algoritmus, ktory v case O(log n) zisti, ci tvoria dobre uzatvorkovany vyraz.

Ja som tam vyrabal prilis zlozite a asi aj nepresne algoritmy, ale ktosi predo mnou to spravil cez Eulerovske tahy. Uzatvorkovanie sa da reprezentovat stromom, cize aj Euler. tahom, ohodnotime hrany smerom dole plus 1, a hore -1. Treba kontrolovat spravnost, napr. cez prefix. sucty zapornost suctu hran a este viacero veci, na ktore si nepamatam.

---

2. 6. 2003

Majme pole A={a_1..a_n} zistite prefixove minimum v case O(log(n)) - cize rychlejsie ako log(n). S linearnym poctom operacii

Najdem minimum v case O(log log(n)), cely tento strom si pamatam a spatne nim prexadzam. Pre kazdy vrchol zmemim jeho minimum na minimum rodica, ak je index minima rodica mensi, ako najmensi prvok intervalu zakoreneneho v danom vrchole

Majme pole A={a_1 .. a_n} z (0,1) a najdite maximalnu vzdialenost a_i-a_j, tak aby neexistovalo k take, ze a_i< a_k <a_j. V case O(log(n))

Nasiel som maximum, minimum a rozdelil to na n+1 intervalov. najst minimum a maximum kazdeho intervalu sa stiha v danych hraniciach. este treba odtial vyhodit prazdne intervaly, co sa da tiez v log(n). (pole nul a jednotiek, spajam iba jednotky). Potom urobim pole (min(i)-max(i-1)) a najdem maximum.

Majme binarny strom T, taky, ze pre kazdy vrchol mame p(v) - rodic a s(v) - surodenec. Zostrojte minimalne pokrytie v O(log (n)). Optimalne.

Najprv treba zistit listy O(1) / easy

Potom pomocou eulerovskeho tahu zoradim vrcholy a v case log(n), z nich vyberiem listy (ide o to aby som ich nejak zoradil). Potom aplikujem standardny algoritmus na vyhodnocovanie vyrazov na binarnych stromoch, funkcia je nand a hodnoty listov 0. Kedze mi v kazdom korku vypadne zhruba polovica vrcholov, zkontrahovat pole vrcholov ide v case O(1).

Mame pole A nul a jednotiek, vypocitat pre kazdu jednotku index jednotky, ktora je od nej z lava najblizsie v case O(log log n) a praci O(n)

Pomocou 2-logaritmickeho stromu...

Na zaciatku mam n postupnosti dlzky 1 tvorenych prvkami pola A, pre kazdu postupnost si pamatam jej dlzku (naz zaciatku 1) a index jej najpravejses jednotky (index je lokalny vzhladom na postupnost, je rovny 0 ak tam nie je ziadna jednotka)..

V kazdom poli si pamatam index najblizsej najlavejsej jednotky.. Potom v kazdom kroku zoberiem 2 postupnosti a zmergujem ich dokopy to mozem urobit v case O(1) tak, ze updatujem indexy v tom druhom poli podla tej hodnoty v prvom poli ... no nie je to dotiahnute ale podrobne riesenie prenechavame na pozorneho citatela :)

Zistite v case O($\log^2$n) ci je graf bipartitny

Najprv si treba spomenut, ze graf je bipartitny <=> neobsahuje kruznicu neparnej dlzky

Vzorove riesenie si zoberie lubovolnu kostru, fixuje jeden vrchol, vypocita Eulerom hlbky ostatnych vrcholov, po urovniach ich ofarbi dvoma farbami a potom pre vsetky mimokostrove hrany kontroluje, ci vedu medzi vrcholmi roznej farby.

Moje riesenie spocivalo vo vypocitani postupnosti matic

A^1, A^2, A^3, ..., A^n kde A^1 je matica susednosti grafu. plati, ze A^k[i,j] = 1 <=> ak existuje medzi i a j cesta dlzky k. cize este treba overit, ci A^k[i,i] = 1, ak k je neparne.

Je zadany lubovolny (aj nekonvexny) n-uholnik a bod p. Skonstruujte optimalny algoritmus s casom O(log n) na zistenie, ci p lezi vnutri n-uholnika.

(Vraj) vseobecne znamy sekvencny algoritmus funguje tak, ze si zoberie lubovolnu polpriamku vychadzajucu z p a zrata, kolkokrat polpriamka pretne strany n-uholnika. ak parny pocet krat, tak bod lezi mimo, ak neparny, tak vnutri.

Zparalelnenie spociva v priradeni procesora kazdej hrane, spocitanie, ci polpriamka z p pretina tuto stranu. ak ano, tak nastavime C[i] = 1, inak C[i] = 0. nakoniec urobime paralelne sumu na C[i] a pozrieme sa na jej paritu.

---

28. 5. 2003

Mame pole A roznych prvkov a pole B, zlozene s 0 a 1. Uloha je najst prefixove sumy prvkov pola A[i_1 < j <= i_2] takych, ze B[i_1]==1 && B[i_2]==1 && B[j]==0 a to cele na EREW v case O(log n) s O(n) operaciami (rozumej A[i < j <k] == A[i+1] * A[i+2] * ... * A[k]

Vytvorim pomocne pole I, zlozene s indexov na prvky B, take, ze B[i] == 1, napr B=(1,0,0,1,0,1) I=(1,3,6) ... ziskame jednoducho pomocou prefixovych sum na poli B (cas O(log n) pri O(n) opp.)

Teraz paralelne spustime hladanie prefixov na A[i<j<k] (j a k su dva susedne indexy v poly I) (toto nam spapka O(log n) casu a O(n) operacii, na standartnom EREW alg. so skript)

Mame pole |A|=n zlozene z 0 a 1. Dokazte, alebo vyvratte, ci sa da najst prva 1 tohoto pola v case O(1) optimalne na common CRCW. (W = O(n) !)

Da sa: cez dvoj logaritmicke stromy spojeny s hladanim minima (O(1) case, O(N2)op.) a s eliminaciou prazdnych usekov...

Navrhnite optimalny algoritmus na vypocet maximalnej suvislej podpostupnosti v poli v case O(log n).

Na pole som postavil strom, binarny, korenom smerom k nebu :)

Mame postupost $A=(a_1, \ldots a_n)$ celych cisiel a pole nul a jednotiek B dlzky n, take ze $b_1=b_n=1$. Pre kazde $i_1<i_2$ take ze $b_{i1}=b_{i2}=1$ a pre vsetky j take ze $i_1<j<i_2$, plati: $b_j=0$ chceme vypocitat prefixove sucty $(a_{i1+1}, \ldots a_{i2})$. Zostrojte alg. s casom O(log n) s O(n) operaciami na EREW PRAM.

1. faza pustim prefix-sum na pole B, mam identifikovane segmenty

2. faza prefix-sum na A s obmedzenim, ze prvky A[i] a A[j] sa scitaju len ak B[i]=B[j]

$\boxed{1}$ Zafarbite vrcholy orientovaneho kruhu 3 farbami v paralelnom case O(log n) a v praci O(n), kde n je dlzka kruhu. Kruh je tvoreny n vrcholmi ocislovanymi 1..n; kazdy vrchol ma jednu vstupnu a jednu vystupnu hranu. Hrany su dane v poli S dlzky n, tak ze S[i]=j ak <i,j> je orientovana hrana v kruhu. Aky PRAM model pouzivate ?

$\boxed{2}$ Su dane dve utriedene postupnosti A a B, obe dlzky $n=k^2$. Napiste algoritmus, ktory spoji A, B do jednej utriedenej postupnosti v case O(log log n) s pouzitim O(n log log n) operacii na CREW PRAM modeli.

$\boxed{3}$ Uvazujte cyklus $C=(v_1,..,v_n)$ a mnozinu hran E medzi vrcholmi z C taku, ze pre kazdy vrchol vi existuje najviac jedna hrana z E incidentna s vi. Problem je urcit, ci sa daju vsetky hrany z E nacrtnut vnutri cyklu C tak, ze sa nepretinaju. Uvedte EREW PRAM algoritmus na riesenie tohoto problemu v case O(log n) a praci O(n).

$\boxed{4}$ Je dana postupnost $A=(a_1,..,a_n)$ a boolovske pole B dlzky n take, ze plati $b_1=b_n=1$. Pre kazde $i_1<i_2$ s vlastnostou $b_{i1}=b_{i2}=1$ a $b_j=0$ pre kazde $i_1<j<i_2$ chceme vypocitat sucet prefixov podpola $(a_{i1+1},..,a_{i2})$. Uvedte algoritmus na vypocet vsetkych zodpovedajucich prefixovych suctov v case O(log n) s pouzitim O(n) operacii na EREW PRAM modeli.

$\boxed{5}$ Nech A je pole s n prvkami z mnoziny S a nech x je z S. Urcte rank(x:A) v paralelnom case O(log n) a v praci O(n) na EREW PRAM modeli.

$\boxed{1}$ Zafarbite vrcholy orientovaneho kruhu dlzky n 3 farbami v paralelnom case O(log*n) a v praci O(n log* n). Kruh je tvoreny n vrcholmi ocislovanymi 1...n; kazdy vrchol ma jednu vstupnu a jednu vystupnu hranu. Hrany su dane v poli S dlzky n tak, ze S[i] = j ak <i,j> je orientovana hrana v kruhu. Aky PRAM model pouzivate?

$\boxed{2}$ Urcite maximum z n prvkov optimalne na common CRCW PRAM v case O(log log n).

$\boxed{3}$ Je dane pole $A = (a_1, \ldots , a_n)$. Lavy zasah prvku a_i je prvok a_k (ak existuje) taky, ze k je maximalny index splnajuci $0<k<l$, $a_k < a_i$. Podobne definujeme pravy zasah prvku a_i. ANSV problem (ANSV – all nearest smaller values) je urcit prave aj lave zasahy vsetkych prvkov z A.

$\boxed{4}$ Nech T = (V,E) je korenovy strom, specifikovany hranami (i,p_i), 1<=i<=|V| = n, kde p_i je predchodca i. Treba ukazat, ze T sa da zafarbit 2 farbami. Uvedte algoritmus, ktory najde 2-zafarbenie stromu T v case O(log n) na CREW PRAMe.

$\boxed{5}$ Uvedte triediaci algoritmus, ktory utriedi n prvkov optimalne v case O(log n log log n) na CREW PRAMe.